

# Measuring the Complexity of Ultra-Large-Scale Evolutionary Systems

Michele Amoretti<sup>1</sup>, Carlos Gershenson<sup>2</sup>

1: Centro Interdipartimentale SITEIA.PARMA, Università degli Studi di Parma

2: Computer Science Department, IIMAS, Universidad Nacional Autónoma de México

**Abstract**—Ultra-large scale (ULS) systems are becoming pervasive. They are inherently complex, which makes their design and control a challenge for traditional methods. Here we propose the design and analysis of ULS systems using measures of complexity, emergence, self-organization, and homeostasis based on information theory. We evaluate the proposal with a ULS computing system provided with genetic adaptation mechanisms. We show the evolution of the system with stable and also changing workload, using different fitness functions. When the adaptive plan forces the system to converge to a predefined performance level, the nodes may result in highly unstable configurations, that correspond to a high variance in time of the measured complexity. Conversely, if the adaptive plan is less "aggressive", the system may be more stable, but the optimal performance may not be achieved.

**Index Terms**—ultra-large-scale system; peer-to-peer; evolution; complexity; information theory

## I. INTRODUCTION

Ultra-large-scale (ULS) systems are the result of the interconnection of heterogeneous systems — characterized by decentralized goals and control — that as a whole exhibit one or more properties (*i.e.* behavior) which are not easily inferred from the properties of the individual parts. ULS are complex systems, since the interactions of their components determine their future state and that of the system [11]. This interconnectedness limits the predictability of ULS, making traditional methods that rely on prediction inadequate [10].

Examples of ULS systems are the Internet, healthcare infrastructures, e-markets, global ambient intelligence systems, and distributed high-performance computing facilities. To overcome the rapidly growing complexity of their management, and to reduce the barrier that complexity poses to further growth, a variety of architectural frameworks based on "self-regulating" components has been proposed [19]. Adaptive plans may turn ULS systems into a more efficient, environment-driven systems, provided that the adaptive plan itself does not introduce further complexity and instabilities [6].

In this paper we propose the evaluation of a distributed evolutionary strategy, by means of the complexity measurement principles introduced by Gershenson and Fernandez in [9].

There are dozens of measures of complexity. Several of them are based on information theory [18]. This is convenient because anything can be measured in terms of information, thus these measures can be applied to any studied phenomenon. Some measures of complexity correlate with

"disorder" or "chaos", making random phenomena to have the highest complexity. However, other approaches consider complexity as a balance between chaos and order. This balance is desirable for computing and living systems, since they require certain stability but also certain variability. The ordered extreme is robust, but does not enable adaptation. The chaotic extreme allows for adaptation and exploration, but information cannot be lost. In this approach, complexity is seen as a balance between propagating and transforming information [8].

The paper is organized as follows: In the following section, related work concerning ULS evolutionary systems is mentioned. In section III, the proposed methodology is presented. In section IV, the case study of an ULS peer-to-peer computing system with distributed genetic adaptation is used as an illustration. In section V, results of the case study evaluation carried out by means of discrete event simulation are shown. Section VI concludes the paper with a summary of achieved results and an outline of future work.

## II. RELATED WORK

The SEI study [17] on ULS systems brings together experts in software and other fields to examine the consequences of rapidly increasing scale in software-reliant systems. The report details a broad, multi-disciplinary research agenda for developing the ultra-large-scale systems of the future, that also include computer-supported evolution, adaptable structure and emergent qualities.

All these aspects can be placed under the umbrella of autonomic computing, that proposes to provide distributed systems with four key properties: self-configuration, self-healing, self-optimization, and self-protection [15]. IBM has suggested a reference model for autonomic control loops, which is sometimes called the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) loop [14]. This model is being widely used to communicate the architectural aspects of autonomic systems. The MAPE-K construction can be iterated, as the control loop itself could be adaptive (see for example [16] and [5]).

With respect to *evolutionary adaptation*, Hales introduced an algorithm called SLAC, which means selfish link and behavior adaptation to produce cooperation [13]. SLAC is based on the copy and rewire approach, whose basic algorithm assumes that peer nodes have the freedom to change the way they handle and dispatch requests to and from other nodes, and drop and make links to nodes they know about. Another

interesting approach for node restructuring has been proposed by Tyson *et al.* [20], that show how survival of the fittest has been implemented into the Juno middleware. On receipt of a superior component, Juno dynamically reconfigures the internal architecture of the node, by replacing the existing component with the new one. The Distributed Remodeling Framework (DRF) [2] is a general approach for the design of efficient environment-driven peer-to-peer networks. Thanks to the DRF, the modifications of the load on the whole system trigger reconfigurations at the level of single peers, from which global system reconfiguration quickly emerges without a centralized control.

### III. METHODOLOGY

The ULS systems we want to characterize are made of several thousands nodes that interact in a peer-to-peer fashion, *i.e.* without a centralized control. Each peer has a modular structure that can dynamically change, by adding or removing components. Being  $p$  the total number of component types for the system, we define a vector  $\mathbf{M}$  of  $q \leq p$  components, called the *model* of the peer. An adaptive plan  $\tau$  produces a sequence of configurations, *i.e.* a trajectory through the *search space*  $\mathcal{M}$  of models, following an evolutionary process. Further details — not necessary for the purposes of this paper — are given in [2].

By means the framework introduced by Gershenson and Fernandez in [9], we measure the evolution over time of the information  $I$  associated to  $\mathbf{M}$ , when the components of the latter are integer values  $x \in [1..n]$  used as parameters for the node's functional processes. Taking into account the latest  $W$  configurations of  $\mathbf{M}$ , we measure the frequency of each value for  $x \in [1..n]$ . Then we use the frequency as  $P(x)$  and compute the normalized information as

$$I = \frac{-\sum P(x) \log_2 P(x)}{I_{max}} \quad (1)$$

where  $I \in [0, 1]$  and  $I_{max} = -\log_2 1/n$ , since the maximum information value is achieved when all values  $1, \dots, n$  have the same probability [9]. Minimum information ( $I = 0$ ) occurs when only one value is repeated in time. This implies that the node is static, *i.e.* there is no change.

For example, if the latest  $W = 3$  configurations of  $\mathbf{M}$  have been  $(1, 3, 5), (1, 3, 6), (1, 4, 6)$ , then the frequencies are  $P(1) = 1/3, P(2) = 0, P(3) = P(6) = 2/9, P(4) = P(5) = 1/9$ . The associated information is

$$I = \frac{-1/3 \log_2 1/3 - 4/9 \log_2 2/9 - 2/9 \log_2 1/9}{-\log_2 1/6} = 0.85$$

Since peers are randomly initialized, we assume a random “input” and use the following simplified formulas [9]:

$$\text{emergence} \quad E = I \quad (2)$$

Higher emergence implies that the process *produces* more information. Low emergence implies that the process does not produce information, *i.e.*  $I = 0$ .

$$\text{self-organization} \quad S = 1 - I \quad (3)$$

In this simplified equation, self-organization can be seen as the opposite of emergence. Low emergence implies high self-organization and vice versa. The most self-organized process is that which is static, while the least self-organized process is that which changes the most, *i.e.* it is the most emergent ( $I = 1$ ).

$$\text{complexity} \quad C = 4 \cdot E \cdot S \quad (4)$$

where the 4 factor is for normalization reasons, since  $I \in [0, 1]$  and  $C$  is maximized when  $I = E = S = 0.5$ . Complexity here is seen as a balance between emergence (change) and self-organization (order). Complexity is low when  $E$  or  $S$  are high, while it is maximal when they are equal.

$$\text{homeostasis} \quad H = 1 - d(I, I_{init}) \quad (5)$$

where  $d$  is the normalized Hamming distance between the  $\mathbf{M}$  associated to  $I$  and  $I_{init}$ , which is the information associated to the initial configuration of the peer.  $H$  is a complementary measure of change in the system. The highest  $H = 1$  is given when there is no change,  $H \approx 1/n$  indicates lack of correlation between compared states.

### IV. CASE STUDY

We use the example of an ultra-large-scale computing system where nodes form a peer-to-peer overlay network, whose global functioning is the result of the interactions among nodes, and depends on their local evolution based on an adaptive plan implemented as a genetic algorithm (GA). In the considered system, shared resources cannot be acquired (by replication) once discovered, but may only be directly used upon contracting with their hosts. An example of such resources is disk space, which can be partitioned and allocated to requestors for the duration of a task, or in general for an arranged time.

The topology of the envisioned ultra-large-scale system can be represented as an undirected graph whose arcs stand for mutual knowledge among peers. The node degree  $k$  is the number of neighbors of each node. Its statistical distribution depends on the history and on the dynamics of the network, and may affect the performance of the distributed algorithms which are executed.

Every peer executes a resource lookup algorithm based on *epidemic* query propagation [7]. In details, peers generate lookup messages that are matched with local resources and if necessary propagated to neighbors. Each peer has a cache that contains the descriptors of remote resources that have been previously discovered. Such a cache is used before sending random (*i.e.* blind) queries. Every lookup message has a time-to-live ( $T$ ), representing the remaining number of hops before the message itself expires — *i.e.* it is no longer propagated.

Moreover, every peer caches received queries, in order to drop subsequent duplicates.

The resource discovery process is affected by three parameters:

- $f_k$  = fraction of neighbors targeted for query propagation
- $T_{max}$  = propagations depth, *i.e.* the maximum number of times a query is forwarded by peers before it is removed from the network
- $D_{max}$  = maximum size of the cache

through *phenotype* of each peer, whose most simple form is:

$$\Phi = G(f_k, T_{max}, D_{max}) = \phi_0 f_k + \phi_1 T_{max} + \phi_2 D_{max} \quad (6)$$

where constant weights  $\phi_0, \phi_1, \phi_2 \in \mathbb{R}$  control the influence of each parameter.

In traditional epidemic algorithms, parameters have fixed values, set in advance according to the results of some tuning process which should guarantee good performance with high probability. Actually, if all peers would be configured with  $f_k = 1$  and the same  $T$  value, the resulting scheme would be exactly the Gnutella protocol [12]. In our scheme, parameters are functions of the chromosome, thus randomly initialized when a peer is created and joins the network. Moreover, the adaptive tuning of parameters is based on a GA with a fitness function  $F$  to be minimized, that depends on two parameters. The first one is the average *query hit ratio*

$$\langle QHR \rangle = AQ(QHR_0, \dots, QHR_k) = \frac{1}{k+1} \sum_{j=0}^k QHR_j \quad (7)$$

The query hit ratio is the number of query hits  $QH$ , *i.e.* successful lookups, versus the number of submitted queries  $Q$ . In the previous equation, we assume that  $j = 0$  for the considered peer, and  $j = 1, \dots, k$  for its neighbors.

All peers are characterized by a genotype  $\mathbf{M}$ , defined by three genes  $\{M_0, M_1, M_2\}$  with values in a limited subset of  $\mathbb{N}$ . The mapping between the phenotype and genotype is given by the following equations:

- $f_k = c_0 M_0$
- $T_{max} = c_1 M_1$
- $D_{max} = c_2 M_2$

where  $c_i \in \mathbb{R}$ , (with  $i = 0, 1, 2$ ) are constants.

#### A. Fitness functions

Within the set of fitness functions we defined in [2], we chose the following:

$$F_2(\Phi, \langle QHR \rangle) = (1 - \langle QHR \rangle) \frac{1}{\Phi} + \langle QHR \rangle \Phi$$

$$F_4(\Phi, \langle QHR \rangle) = \left( \frac{\langle QHR \rangle}{Th} - 1 \right) \left( \frac{\Phi}{\Phi_M} \right) + (1 - \langle QHR \rangle)$$

where  $Th$  is a threshold value and  $\Phi_M = \max\{\Phi\}$ .

Both fitness functions reward lower phenotypes when the average query hit ratio  $\langle QHR \rangle$  is high, and higher ones when the  $\langle QHR \rangle$  is low.  $F_4$  allows to set a threshold  $Th$  for  $\langle QHR \rangle$ , under which the  $M_i$  are forced to grow. Conversely, when  $\langle QHR \rangle > Th$  the  $M_i$  are forced to decrease.  $F_2$  does not allow to set a threshold for  $\langle QHR \rangle$ .

#### B. Adaptive plan

The block diagram in Fig. 2 illustrates how each peer works. The resource lookup is executed in a separate thread with respect to the adaptation process. However, they influence each other.

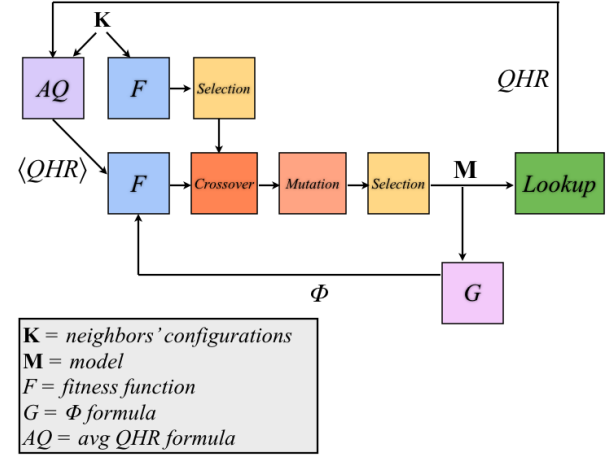


Fig. 2. Functional architecture of the peer.

#### Algorithm 1 Adaptation

```

1: let  $g = 0$ 
2: while not converged do
3:   evaluate the fitness of own model
4:    $g = g + 1$ 
5:   select neighbor with best fitting model
6:   perform cross-over with best neighbor to generate offspring  $\mathcal{O}_g = \{\bar{\mathcal{O}}_{g1}, \bar{\mathcal{O}}_{g2}\}$ 
7:   mutate offspring in  $\mathcal{O}_g$  with probability  $1 - \langle QHR \rangle$ 
8:   select the new generation  $\mathcal{M}_g$  from the previous generation  $\mathcal{M}_{g-1}$  and the offspring  $\mathcal{O}_g$ 
9: end while

```

The pseudocode in Algorithm 1 describes the adaptive plan which is periodically executed by each peer. The best neighbor is chosen using proportional selection, *i.e.* the chance of the neighbors' models to be selected is inversely proportional to their fitness values. Cross-over is always performed, with a randomly-generated crosspoint. Mutation depends on  $\langle QHR \rangle$ , being highly improbable while the average query hit ratio of the peer and its neighbors tends to 1. The final selection between current peer's chromosome and the mutated offspring is random, with probability that is inversely proportional to their fitness values.

#### V. SIMULATIONS

To evaluate the performance of the proposed ultra-large-scale system, with and without GA-based adaptation, we used a general-purpose discrete event simulation environment, called DEUS, based on Java and XML, released as open source under the GPL license [1]. Such a tool has been used instead of classical network simulators (*e.g.*, ns-2 or Opnet), since it is optimized to analyze P2P networks at a higher level. In particular, with DEUS it is possible to simulate highly dynamic

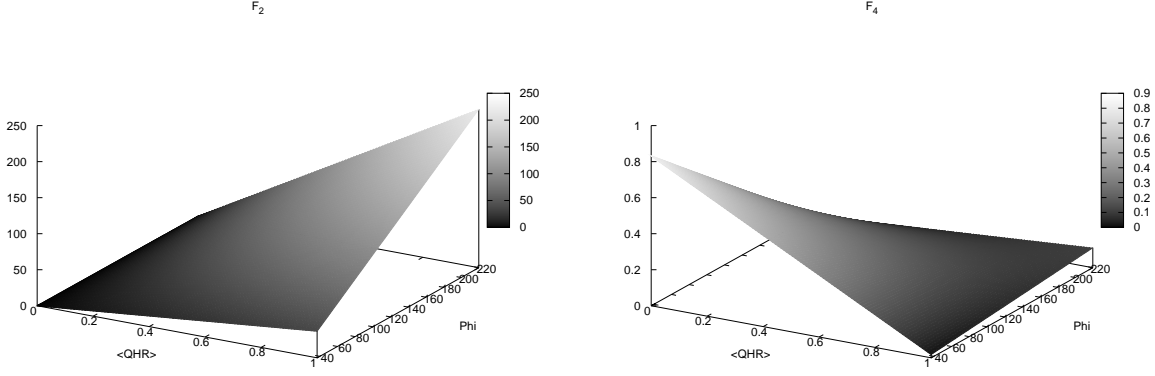


Fig. 1. Plot of  $F_2$  and  $F_4$  with the parameters' values of the simulation experiments.

overlays (*i.e.* application level) networks, with several hundred thousands nodes, on a single machine — without the need to simulate also lower network layers (whose effect can be taken into account, in any case, when defining the virtual time scheduling of message propagation events).

Each simulated peer was characterized by three kinds of consumable resources: CPU, RAM, and disk space. Their values were randomly generated (with uniform distribution) as multiple of, respectively, 512 MHz, 256 MB, and 10 GB. The maximum amount of resources per peer was 2 GHz, 1 GB for the RAM, and 100 GB for the disk space.

We assumed  $M_i \in [1..6]$  ( $n = 6$ ), and

- $f_k = M_0/6$
- $T_{max} = M_1$
- $D_{max} = 2M_2$

The phenotype  $\Phi$  is given by eq. 6, with  $\phi_0 = 100$ ,  $\phi_1 = 10$ , and  $\phi_2 = 5$ , for which  $f_k$  has more importance than  $T_{max}$  and  $D_{max}$  in the computation of the fitness value. The average query hit ratio  $\langle QHR \rangle$  is given by eq. 7. We assumed that each peer at the beginning has  $QHR = 0.5$ .

All the experiments were carried out with a simulated P2P network of  $N = 10000$  nodes, with a topology randomly constructed without preferential attachment, starting from  $N_0$  completely connected nodes, and each other peer being attached to  $m \in [1, N_0]$  existing peers, with even probability. All connections are bidirectional, *i.e.* if node  $n_a$  has node  $n_b$  in its peerview, then  $n_b$  has  $n_a$  in its peerview. The resulting degree distribution is exponential [4]:

$$P(k) = (1 - e^{-\frac{1}{m}})e^{(1-\frac{k}{m})} \forall k \geq m$$

with expected value

$$E(k) = \sum_{k=m}^{\infty} kP(k) = \frac{e^{(1-\frac{1}{m})}}{1 - e^{-\frac{1}{m}}}$$

In these experiments, we used  $m = 3$  and  $N_0 = 5$ .

Every node performs adaptation every 7 seconds. Figures 3 to 6 illustrate the average evolution of  $M$  and  $QHR$  over

time, for each considered fitness function. We considered two scenarios: stable load and variable load.

In the first scenario, we simulated 150 minutes of the life of a network of peers, with resource queries occurring continuously and independently of one another — according to a Poisson process with rate  $\lambda = 0.028 \text{ s}^{-1}$ , *i.e.* 36 queries every second, each one associated to a randomly chosen node. Each query is a request for a randomly generated amount of resources (no more than 2 GHz of CPU, 1 GB of RAM, and 100 GB of disk space, respectively). Once a resource has been found, it is consumed for a random time interval (according to an exponential distribution with mean value 280 s — for which the system's utilization would be 1, if a provider was found for every request).

In the second scenario, we still simulated 150 minutes of the life of a network of peers, but we considered a load on system changing over time. In the first 50 minutes, almost all resource queries ask for 2 GHz, 1 GB of RAM, and 100 GB of disk space (like in previous experiments). In the following 100 minutes the load is reduced, with almost all resource queries asking for 256 MHz, 128 MB of RAM, and 1 GB of disk space.

We measured also the evolution over time of the information  $I$  associated to the current genotype  $M$ , according to the procedure illustrated in section III. Every 10 virtual minutes, the simulator executes a log event, where peers that have performed at least one lookup are considered for computing the mean value and standard deviation of  $M_i \forall i \in [0, 2]$ ,  $QHR$ ,  $E$ ,  $S$ ,  $C$  and  $H$ . Figures 7 to 10 are related to the case of  $W = 1$ .

For the stable load scenario,  $F_2$  produces balanced change in time, reflected in medium  $E$  and  $S$  and a high  $C$ .  $F_4$  offers less change on average, as seen by a higher  $S$  and  $H$ , lower  $E$ , and medium  $C$ . Still, the rate of information change changes considerably in time, leading to a variance in the measures, especially high for  $C$ .

For the variable load scenario,  $F_2$  performs in a similar fashion: it is not able to adapt to the reduction of the demand. Conversely,  $F_4$  does adapt at the demand change at minute

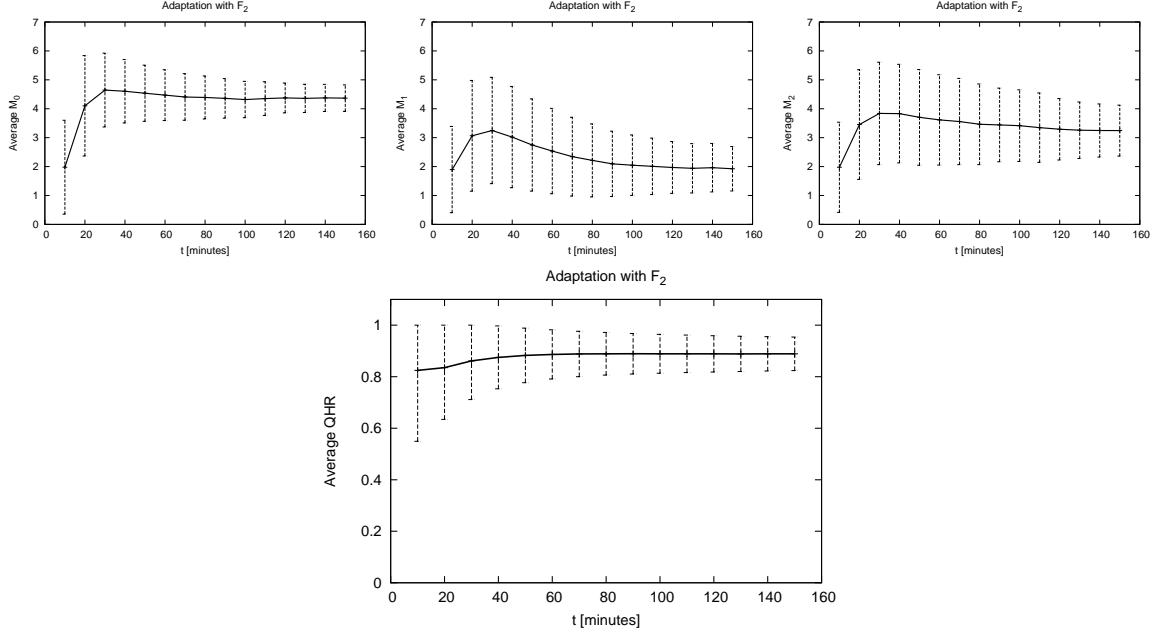


Fig. 3. Average  $M_0$ ,  $M_1$ ,  $M_2$  and  $QHR$  over time, for fitness function  $F_2$ . The load on the system is stable.

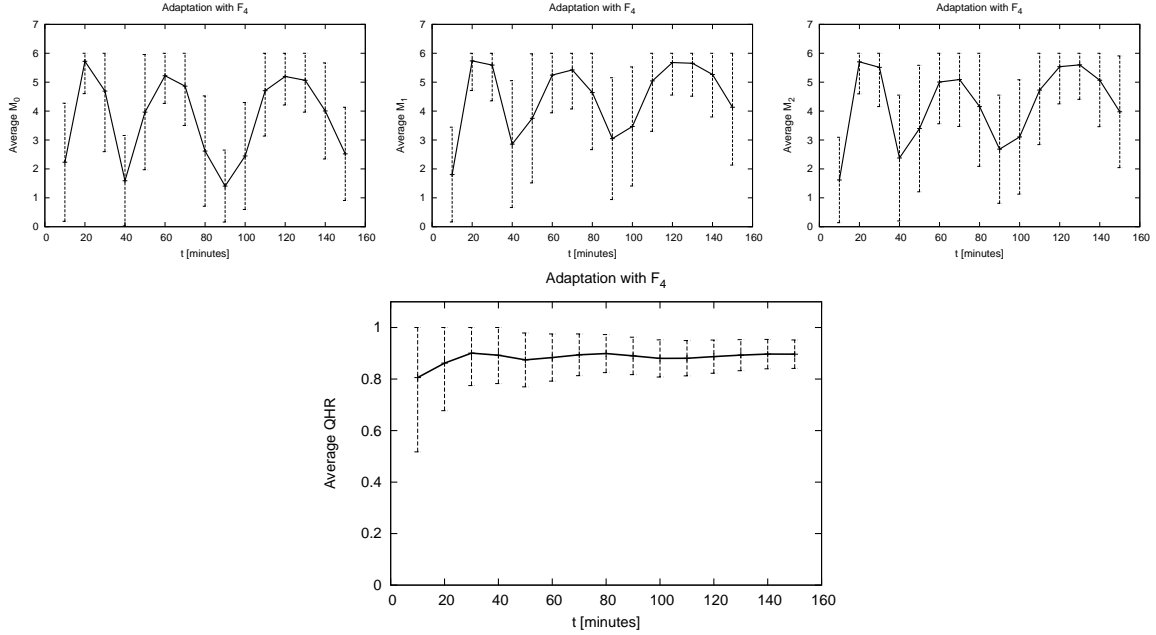


Fig. 4. Average  $M_0$ ,  $M_1$ ,  $M_2$  and  $QHR$  over time, for fitness function  $F_4$ . The load on the system is stable.

50, stabilizing at minute 80. This is seen in maximal  $S$  and  $H$  and minimal  $E$  and  $C$ , indicating that there is no change in the nodes. This is because of the threshold used in  $F_4$ .

## VI. CONCLUSION

We presented simulation results of an ULS computing system with two different types of genetic adaptation while measuring their complexity, emergence, self-organization and homeostasis. The main result is that less "aggressive" adaptive

plans lead to a more stable system, but the optimal performance may not be achieved.

Future work will follow two main directions. Firstly, with respect to the use case proposed in this paper, we will study the impact of parameter variations and we will consider also other workload profiles. Then, we will generalize the analysis to the case of systems with components that dynamically change their structure, by removing or adding components.

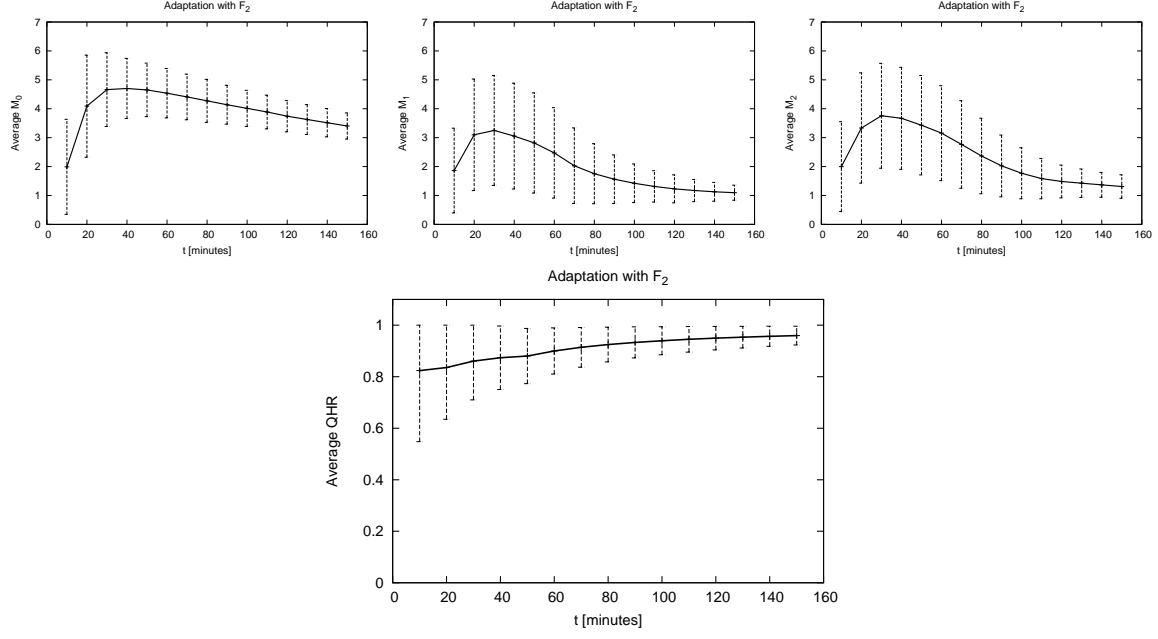


Fig. 5. Average  $M_0$ ,  $M_1$ ,  $M_2$  and  $QHR$  over time, for fitness function  $F_2$ . The load on the system changes at minute 50.

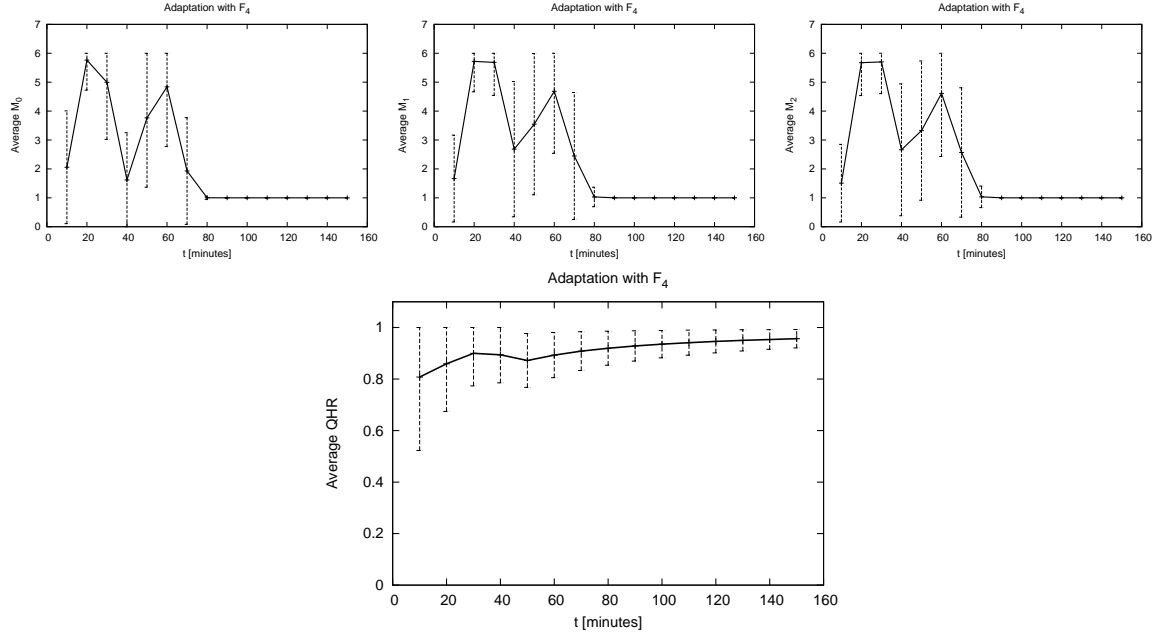


Fig. 6. Average  $M_0$ ,  $M_1$ ,  $M_2$  and  $QHR$  over time, for fitness function  $F_4$ . The load on the system changes at minute 50.

## REFERENCES

- [1] M. Amoretti, M. Agosti, F. Zanichelli, DEUS: a Discrete Event Universal Simulator, to appear in the Proc. of the 2nd ICST/ACM International Conference on Simulation Tools and Techniques (SIMUTools 2009), Roma, Italy, March 2009.
- [2] M. Amoretti, *Distributed Remodeling Framework*, submitted to Artificial Intelligence, ed. Elsevier, 2012.
- [3] J.-P. Banatre, Y. Radenac, P. Fradet, *Chemical Specification of Autonomic Systems*, Proc. 13th Int'l Conference on Intelligent and Adaptive Systems and Software Engineering, Nice, France (2004).
- [4] A.-L. Barabasi, R. Albert, *Emergence of Scaling in Random Networks*, Science, Vol.286, 15 October 1999.
- [5] R. Bruni, A. Corradini, F. Gadducci, A. Lluch Lafuente, A. Vandin, *A Conceptual Framework for Adaptation*, Proceedings of the 15th Int'l Conference on the Fundamentals of Software Engineering (FASE'12), LNCS, vol. 7212, pp. 240254, Springer (2012).
- [6] S. Dobson, R. Sterritt, P. Nixon, M. Hinchey, *Fulfilling the Vision of Autonomic Computing*, IEEE Computer Magazine (2010).
- [7] P.T. Eugster, R. Guerraoui, A.-M. Kermarrec, L. Massoulie, *From Epidemics to Distributed Computing*, IEEE Computer, Vol. 37, No. 5, May 2004.
- [8] C. Gershenson, *The World as Evolving Information*, International Conference on Complex Systems ICCS2007, <http://arxiv.org/abs/0704.0304>.
- [9] C. Gershenson and N. Fernandez, *Complexity and Information: Measuring Emergence, Self-organization, and Homeostasis at Multiple Scales*,

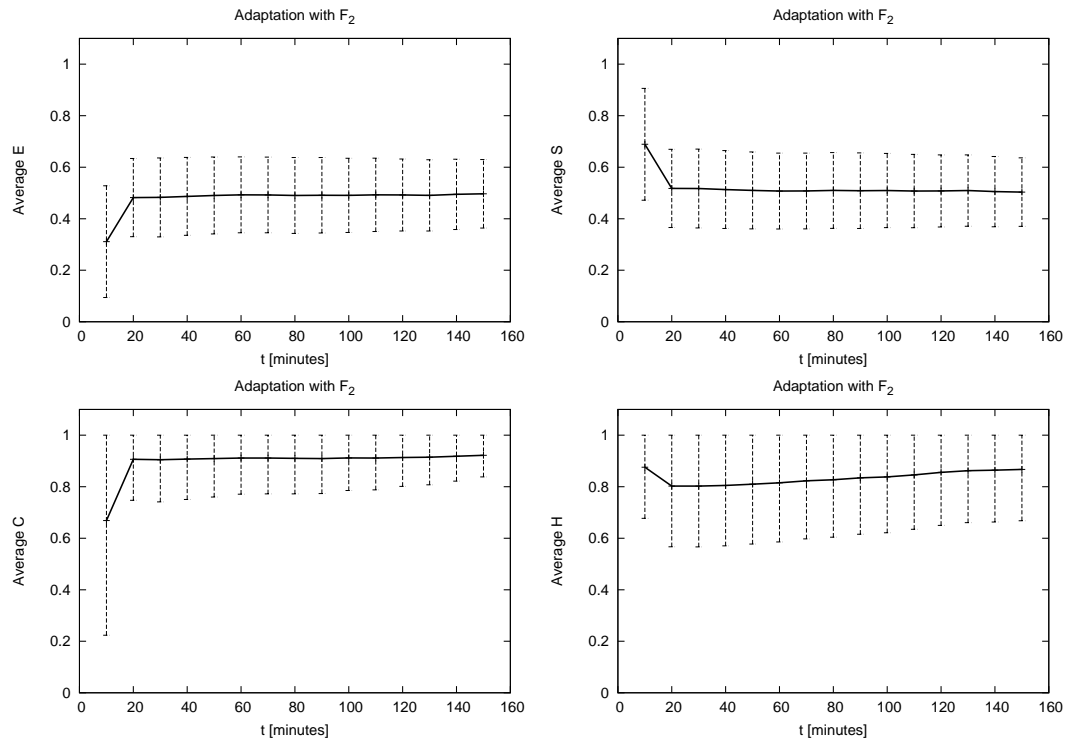


Fig. 7. Average  $E$ ,  $S$ ,  $C$  and  $H$  over time, for fitness function  $F_2$ . The load on the system is stable.

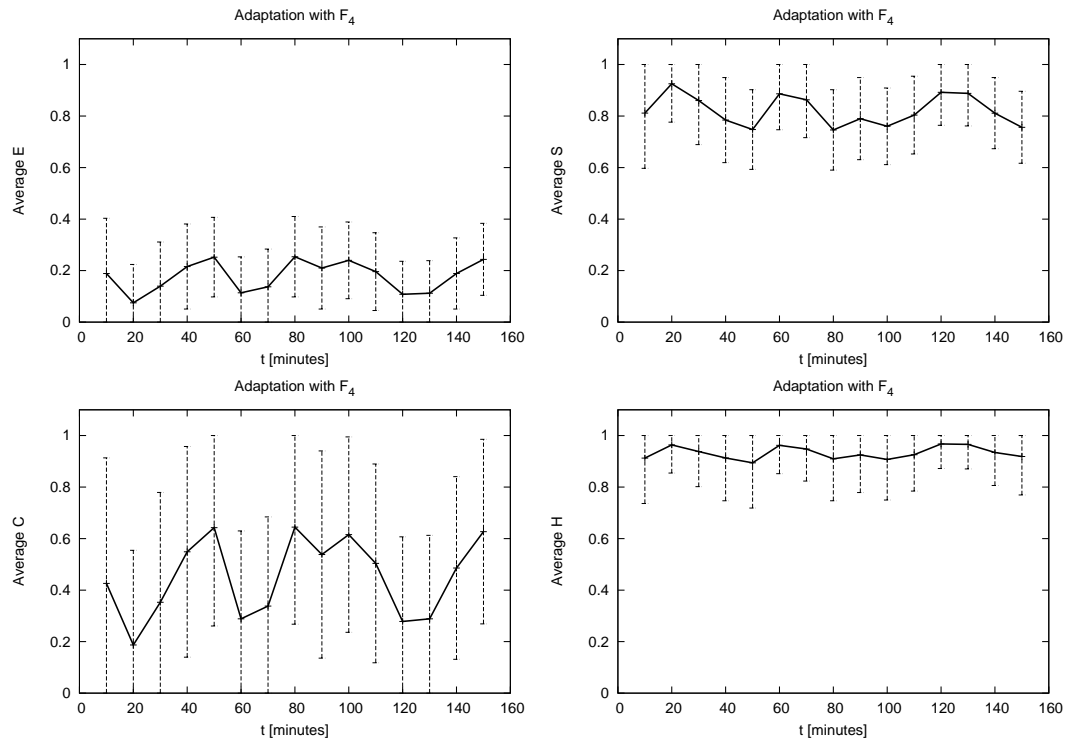


Fig. 8. Average  $E$ ,  $S$ ,  $C$  and  $H$  over time, for fitness function  $F_4$ . The load on the system is stable.

C3 Technical Report, 2012.03. <http://arxiv.org/abs/1205.2026>.

[10] C. Gershenson, *The Implications of Interactions for Science and Philosophy*, Foundations of Science, In Press. <http://arxiv.org/abs/1105.2827>

[11] C. Gershenson and F. Heylighen *How Can We Think the Complex?*, In

K. Richardson (Ed) *Managing Organizational Complexity: Philosophy, Theory and Application*, Information Age Publishing, 2005, pp. 47–61. <http://arxiv.org/abs/nlin.AO/0402023>

[12] Gnutella Protocol Specification 0.4, <http://rfc->

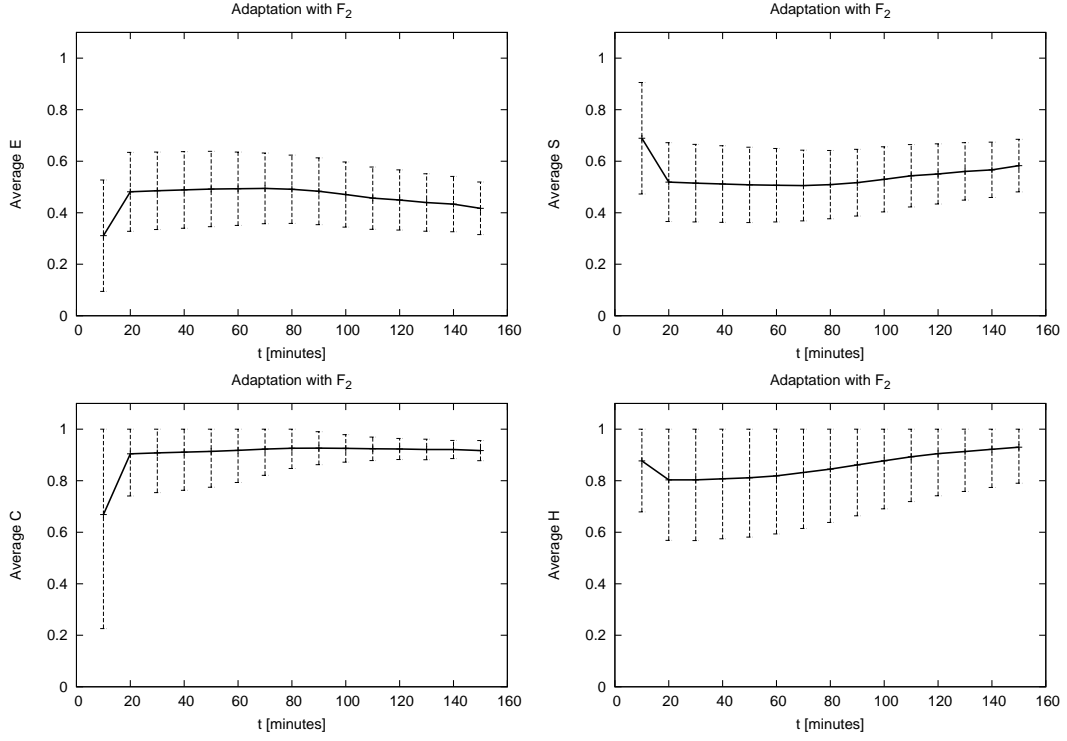


Fig. 9. Average  $E$ ,  $S$ ,  $C$  and  $H$  over time, for fitness function  $F_2$ . The load on the system changes at minute 50.

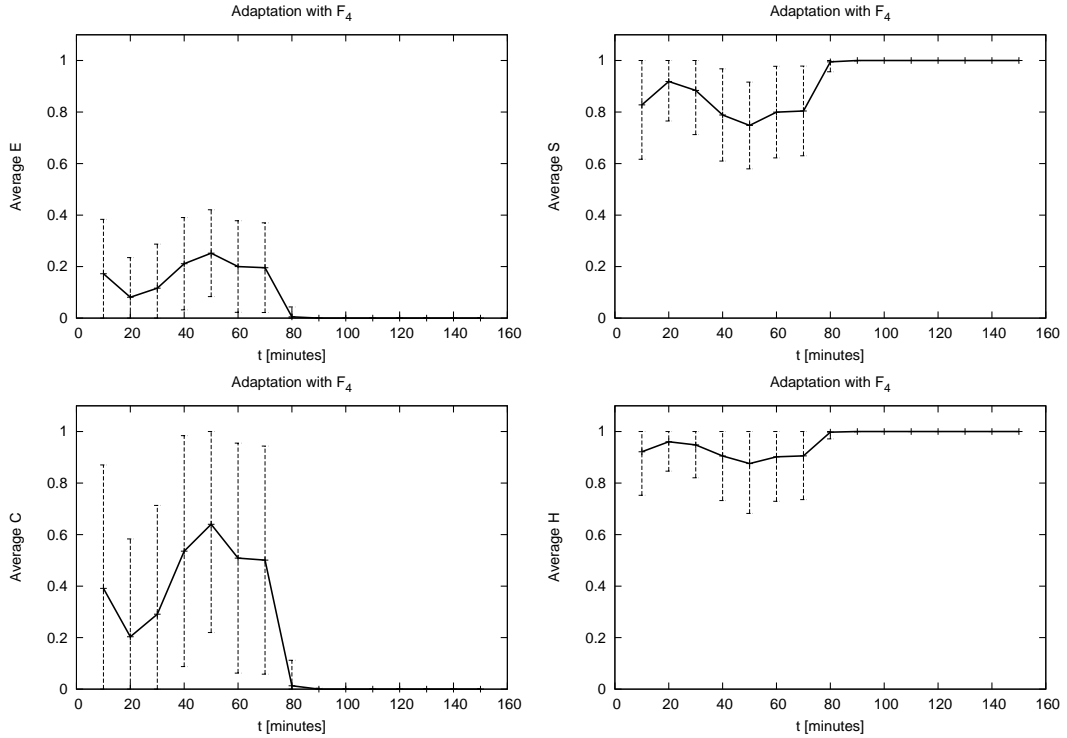


Fig. 10. Average  $E$ ,  $S$ ,  $C$  and  $H$  over time, for fitness function  $F_4$ . The load on the system changes at minute 50.

[gnutella.sourceforge.net/developer/stable/index.html](http://gnutella.sourceforge.net/developer/stable/index.html)

- [13] D. Hales, *From Selfish Nodes to Cooperative networks - Emergent Link-based incentives in Peer-to-Peer Networks*, Proc. 4th IEEE Intl Conf. on Peer-to-Peer Computing, Zurich, Switzerland (2004).

- [14] M. C. Huebscher, J. A. McCann, *A survey of autonomic computing — degrees, models, and applications*, ACM Computing Surveys, Vol. 40, No. 3 (2008).

- [15] IBM, *An architectural blueprint for autonomic computing*. Tech. rep.



- (2003).
- [16] C. Müller-Schloer, H. Schmeck, *Organic Computing - Quo Vadis?*, in Christian Müller-Schloer, Hartmut Schmeck, and Theo Ungerer, editors, *Organic Computing — A Paradigm Shift for Complex Systems*, chapter 6.2, Birkhäuser Verlag, 2011.
  - [17] L. Northrop et al., *Ultra-Large-Scale Systems: The Software Challenge of the Future*, Carnegie Mellon Software Engineering Institute, Ultra-Large-Scale Systems Study Report (2006).
  - [18] M. Prokopenko, F. Boschetti and A. J. Ryan *An Information-Theoretic Primer On Complexity, Self-Organisation And Emergence*, *Complexity*, 15(1):1128.
  - [19] H. Psailer, S. Dustdar, *A survey on self-healing systems: approaches and systems*, *COMPUTING*, Vol. 91, No. 1, pp. 43–73, 2011.
  - [20] G. Tyson, P. Grace, A. Mauthe, S. Kaune, *The Survival of the Fittest: An Evolutionary Approach to Deploying Adaptive Functionality in Peer-to-Peer Systems*, Proc. of the 7th workshop on Reflective and adaptive middleware, Leuven, Belgium (2008).
  - [21] M. Viroli, F. Zambonelli, *A biochemical approach to adaptive service ecosystems*, *Information Sciences* 180 (2010), 1876-1892.